Attorney Docket No. APL1P272D1/P3061D1

PATENT APPLICATION

ACCESSING DIGITAL MEDIA

Inventors:

Amandeep Jawa
225A Dolores St.
San Francisco, CA 94103
Citizen of U.S.A.

Jeffrey L. Robbin
705 Benvenue Avenue
Los Altos, CA 94024
Citizen of U.S.A.

David Heller
2016 Jonathan Avenue
San Jose, CA 95125
Citizen of U.S.A.

Assignee:

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014

Prepared by:

BEYER, WEAVER & THOMAS, LLP
P.O. Box 778
Berkeley, CA 94704-0778

ACCESSING MEDIA ACROSS NETWORKS

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a divisional of co-pending U.S. Application 10/423,638 entitled "ACCESSING MEDIA ACROSS NETWORK," filed on April 25, 2003 from which priority under 35 U.S.C. §120 is claimed, and which is incorporated herein in its entirety and for all purposes.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The present invention relates to digital media and, more particularly, to accessing digital media across networks.

Description of the Related Art

[0003] The ability of computers to be able to share information is of utmost importance in the information age. Networks are the mechanism by which computers are able to communicate with one another. Generally, devices that provide resources are called servers and devices that utilize those resources are called clients. Depending upon the type of network, a device might be dedicated to one type of task or might act as both a client and a server, depending upon whether it is giving or requesting resources.

[0004] Increasingly, the types of resources that people want to share are entertainment-related. Specifically, music, movies, pictures, and print are all types of entertainment-related media that someone might want to access from across a network. For example, although a music library may reside on a family computer in the den, the media owner may want to listen to the music in the living room.

[0005] However, sharing media data can be a network-intensive process. People have devoted significant resources to both reducing the load on networks and increasing the capability of networks to handle large data transfers. Due to advances in compression technology and network bandwidth, the throughput of information through networks has increased dramatically over the years.

[0006] Although the described technologies work well in many applications, there are continuing efforts to further improve the ability to transfer digital media.

SUMMARY OF THE INVENTION

[0007] The present invention provides a method of retrieving media across a network. First, a client connects to a network that includes a server. The server includes at least one media database that has media and associated media information. The client then queries the server for at least a portion of the media information and then receives media information responsive to the query. The client then uses a client-side media management system to manage the received media information. Management of the received media information includes selecting media. The client then requests the selected media from across the network and, in response to the request, receives the requested media.

[0008] In another aspect, a client queries the server for server information and capabilities after connecting to the network. The client then receives a response that identifies the server and informs the client as to its capabilities. After receiving the server information, the client queries the server for database enumeration and receives a response that enumerates all databases, how much media is available, and how many media collections are available. After the database identification, the client queries the server for an enumeration of media collections in the database and receives a response that identifies media collections. The client then queries the server for data associated with an identified media collection, the query being capable of requesting a different level of detail than would be given by default. The response to the media collection query identifies data associated with the identified media collection in the requested level of detail. The client then executes the identified media collection , requesting media from the server when the media collection requires the media and receiving the requested media.

[0009] In yet another aspect, the invention provides a method of ensuring that a media database representation on a client is current. The server first provides a media database that updates to a current revision indicator whenever the media database is modified. Then, the server receives a request from the client, the request pertaining to the database that includes a client-provided revision indicator. After receiving the request, the server compares the current revision indicator to the client-provided revision indicator. The server then responds to the request with a response that includes at least an identification of the current revision indicator if the client-provided revision indicator did not match the current revision indicator.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

Figure 1 is a block diagram illustrating an exemplary environment in which the present invention may be implemented;

Figure 2 is a block diagram illustrating an organizational structure of a server-side media management system on the server illustrated in figure 1;

Figure 3 is a block diagram illustrating an organizational structure of a client-side media management system on the client illustrated in figure 1;

Figure 4A is a representational control flow diagram illustrating one technique that can be used to determine the features of the server-side media management system illustrated in figure 2;

Figure 4B is a representational control flow diagram illustrating one technique that could be used to enumerate databases of the server-side media management system illustrated in figure 2;

Figure 4C is a representational control flow diagram illustrating one technique that could be used to populate a song records portion of the client-side media management system illustrated in figure 5;

Figure 4D is a representational control flow diagram illustrating a technique that could be used to enumerate playlists of the server-side media management system illustrated in figure 2;

Figure 4E is a representational control flow diagram illustrating a technique that could be used to populate a playlist records portion of the client-side media management system illustrated in figure 6;

Figure 4F is a representational control flow diagram illustrating a technique that could be used to retrieve a song from a song database once a song is selected from the client-side media management system illustrated in figure 7;

Figure 5 is a block diagram illustrating an organizational structure of the client-side media management system after receiving a reply to the SERVER-DATABASE request illustrated in figure 4B; Figure 6 is a block diagram illustrating an organizational structure of the client-side media management system after receiving a reply to the DATABASE-SONGS request illustrated in figure 4C;

Figure 7 is a block diagram illustrating an organizational structure of the client-side media management system after receiving a reply to the PLAYLIST-SONGS request illustrated in figure 4E;

Figure 8 is a representational control flow diagram illustrating one technique that could be used to ensure the client and the server illustrated in figure 1 are synchronized; and

Figure 9 is a diagram illustrating an exemplary computing device in which various embodiments of the invention may be implemented.

It is to be understood that, in the drawings, like reference numerals designate like structural elements. Also, it is understood that the depictions in the figures are not necessarily to scale.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0010] In the following description, numerous specific details are set forth to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order to avoid unnecessarily obscuring the present invention.

[0011] The present invention generally allows for client machines to access a media database on a server. Both data and metadata on the server describes and organizes the media in various ways and allows the server to manipulate the media. Instead of relying on the server to execute the media management system, the client requests the data and metadata and then uses the information on a local media management system, effectively creating a representation of the server on the client. Once media is selected on the client through the client-side media management system, the client can request the media from the server, and the server can deliver the media to the client.

[0012] The invention can support both "thick" and "thin" clients. Thick clients are typically software programs, such as iTunes^{TM} software available from Apple Computer, Inc. of Cupertino, CA, operating on hardware devices that support full user interface abilities and have considerable processor and memory resources. Thin clients are software programs operating on hardware devices that may have limited user interface abilities and have reduced processing and memory resources. The invention allows for robust control features appropriate for thick clients, but can adapt to minimal control features for thin clients.

[0013] Generally, a client will first make a request to determine whether media is available on a server. Then, the client can make a series of requests to create a representation of the media available on the server on the client. The representation contains information about the media available on the server ("media information"). Thick clients can choose to retrieve a complete representation of the server's available media, while thin clients may choose to retrieve a partial representation of the server's available media. After receiving the media information from the server, the client (as instructed by its user) can then search, browse, sort, or otherwise interact with the media information now resident on the client. Further, the client (as instructed by its user) will typically select a media item to be presented (e.g., played). In such a case, the media content for the selected media item is streamed from the server to the client.

[0014] In addition, clients can receive notifications from the server when a media database has been changed. Multiple connections can allow a client to use one connection to access media while using another connection to wait for a notification that the database has changed, or to browse media listings.

[0015] Figure 1 is a block diagram illustrating an exemplary environment in which the present invention may be implemented. A network 105 couples a server 110 to various clients 115, 120, 125, and 130. The network 105 can generally be a data network, such as a LAN, WAN or the Internet. The server 110 may or may not be a dedicated device. In the example shown in figure 1, the server 110 is a general purpose computer. The various clients 115, 120, 125, and 130 can be thick or thin clients, with varying levels of processing power. Clients may include portable computers 115, desktop computers 120, specialized devices such as iPods™ 125 available from Apple Computer, Inc. of Cupertino, CA, or even network-aware audio/video components 130 that are designed to work across a network 105. The following discussion will, for simplicity, assume only the portable computer client 115 is requesting information from the server 110.

[0016] Figure 2 is a block diagram illustrating an organizational structure of a server-side media management system 200 on the server 110. The server-side media management system 200 includes a media manager 210 and a music database 205. The media manager 210 controls access to the music database 205. More particularly, the media manager 210 receives requests from the client 115, accesses the music database, and returns responses to the client 115.

[0017] The music database 205 has a number of records 215 and 220 that are used to classify, identify and/or describe media (i.e., media items) in the music database 205. For simplicity, the following discussion will assume the digital media contained on the server 110 only contains music files that can be streamed over the network 105. It should be appreciated that any reference to "songs" or "music" made in this document could be generalized to any form of digital media, which can include sound files, picture data, movies, text files or any other types of media that can be digitally stored on a computer. Similarly, any reference to "playlists" can be generalized to media collections, including collections of mixed digital media.

[0018] The media manager 210 has or can obtain information about the database 205 that may, for example, include the name of the server, the version of the database being used, the type of security that is required, the number of databases available to the server, whether non-standard content codes are supported, whether persistent identification is supported, etc. It should be appreciated by those skilled in the art that the information about the database 205 may exist in a single record file or can be either partially or fully generated on demand, identifying the various pieces of information as needed.

[0019] The song records 215 contain metadata about each media item available in the database 205. The metadata might include, for example, the names of songs, an identification number, a persistent identification number, the artist, the album, the size of the song, the format of the song, the required bit rate, and any other appropriate information. Of course, the type of information may depend on the type of media. A video file might additionally have director and producer fields, but may not use the album field. Still pictures would have no need for bit rate information. While some fields may be standard, others may be specific to certain applications. For example, a video signal may have secondary audio program (SAP) information. A mechanism for ensuring clients can properly process non-standard content codes is described in connection with figure 4A.

[0020] Both an identification number and a persistent identification number can be used. If supported, a persistent identification could be used to access the same information across server restarts. Typically, a server would assign each record a new identification number every time the media management system 200 restarted. However, persistent identification numbers would remain the same for as long as the record is available.

[0021] The playlist records 220 contain information about each playlist available in the music database 205. Further, the information for a given playlist can include the identification numbers for each of the songs within the playlist. Playlists are collections of media that may or may not be in any particular order. Users may choose to combine media by genre, mood, artists, audience, or any other meaningful arrangement. While the playlists 220 on the server 110 will usually only include media contained in its own music database 205, there is no reason the playlist records 220 cannot include media or playlists stored on other servers. However, certain non-standard content codes may need to be used, depending upon the implementation of the server-side media management system 200.

[0022] Figure 3 is a block diagram illustrating an organizational structure of a client-side media management system 300 on one of the clients 115. The client-side media management system 300 includes a media manager 305. The media manager 305 interacts with the media manager 210 of the server-side media management system 200 through the network 105 so as to replicate at least a portion of the music database 205 at the server 110 on the client 115. When the client-side media management system 300 first starts, it cannot access media on the server 110 because it does not as yet have any information about what media is available.

[0023] Figure 4A is a representational control flow diagram illustrating one technique that can be used to determine the features of the server-side media management system 200. Operations performed by the client 115 and the server 110 are represented by corresponding vertical lines 403 and 406. At 409 the client 115 connects to the network 105 and first becomes aware of the server 110. The client 115 can use any connection mechanism that allows it to interact with the network 105. For example, if the client 115 were an iBook™, available from Apple Computer, Inc. of Cupertino, it might use Rendezvous™ networking technology, also available from Apple Computer, Inc. of Cupertino, CA, in order to automatically configure itself with the network 105. If the client is not aware of the server 110, other mechanisms can be used. For example, a user might manually search for the server 110, or the user might directly enter the address of the server 110.

[0024] Once the client 115 is aware of the server 110, it can send a SERVER-INFO request to the server 110 at 412. The SERVER-INFO request is usually used to obtain information from the server prior to attempting any other transactions. If the network 105 uses the TCP/IP protocol, the request could be formatted as an HTTP GET request. The GET request might also allow for additional extensions to be added to the request, enabling, for example, the client 115 to include information about the client-side media management system 300.

[0025] At 415 the server responds to the SERVER-INFO request with information describing a series of features supported by or required by the server. The information might, for example, include information about the server-side media management system 200, the number of available databases, whether and what login procedures are required, whether updates are supported, whether persistent identification numbers are supported, whether non-standard content codes are supported, and the protocol version.

[0026] The information provided to the client 115 at 415 permits the client-side media management system 300 to understand the capabilities of the server 110. Although the client 115 is able to identify the server 110, the client 115 does not yet have any information about the available media.

[0027] If the client 115 determines that the server 110 responded to the SERVER-INFO request with an indication that non-standard content codes are supported at 418, the client 115 can optionally issue a CONTENT CODE request at 421. The CONTENT CODE request is one mechanism by which the client 115 can obtain a list of content codes supported by the server 110 and associated string names mapped thereto.

[0028] The inclusion of the string name allows multiple developers to use the same codes for their individualized products. For example, one developer may assign the code "16000" to a feature that allows users to purchase corresponding albums over the network; while another developer may assign the same code to feature that provides users with the lyrics of songs that are being listened to. By allowing a string name to be included, the client 115 can determine whether it can support the content code. Uniqueness of the string name can, for example, be ensured by including the developer's URL as part of the string name.

[0029] At 424 the server 110 responds to the CONTENT CODE request the codes and their associated string names. At 427, the client 115 can simply ignore the code/string pairs that it does not recognize. Otherwise, for those code/string pairs that the client 115 does recognize, the client 115 will associate the code with the associated string name.

[0030] At 430 the client 115 logs into the server 110. The login procedure might require a user name (or account name) and password so the user of the client can be authenticated. The login procedure is only required if the server 110 requires it. Certain security protocols might require that every future request made by the client 115 include certain parameters such as a session identification number.

[0031] Once logged in, the client 115 is ready to begin populating its local representation of the music database 205. Figure 4B is a representational control flow diagram illustrating one technique that can be used to enumerate databases of the server-side media management system 200. Operations performed by the client 115 and the server 110 are represented by corresponding vertical lines 403 and 406. At 433 the client 115 can issue a SERVER-DATABASE request, which can be used to retrieve the list of all music databases from the server 110. The SERVER-DATABASE request may additionally include an index range and/or a query. Although available to both thick and thin clients, thin clients might use index ranges and/or queries to limit the amount of data contained in each server response.

[0032] The index range might be used in any request to constrain the items returned in the response to a subset of the total set of items, based on the position (or index) of the first item and the number of items requested. For example, an index range could be used to request: the second music database from a server, songs 10 through 20 from a music database, the last 5 playlists from a music database, the first 5 songs from a given playlist, or the 42nd song in a music database.

[0033] The query might be used in any request to constrain the items returned in the response to a subset of the total set of items, based on the specified criteria. For example, a query could request: songs in a database after a given year; playlists that contain a certain word in their name; songs in a database that do not contain a given word in their name; or some combination thereof.

[0034] After processing the SERVER-DATABASE request at 436, the server 110 issues a response at 439. If no index range and/or query was given, the response would contain a complete list of the music databases available at the server 110 together with information about the one or more music databases. The information about each database might, for example, include the database identification number, the persistent database identification number, the name for each database, the numbers of songs, and the number of playlists. With this information, the client-side media management system 300 becomes aware of the general structure of the one or more music databases at the server 110.

[0035] Figure 5 is a block diagram illustrating the organizational structure of the client-side media management system 300 shown in figure 3 after receiving a reply to the SERVER-DATABASE request. At 442, the client 115 is able to identify the music database 510, the number of available song records 515, and the number of available playlists 520. Once the general structure of the music database 510 is known, the client 115 can populate its representation using any number of techniques.

[0036] Figure 4C is a representational control flow diagram illustrating one technique that could be used to populate the song records 515 portion of the client-side media management system 300 for a specific database 510. Operations performed by the client 115 and the server 110 are represented by corresponding vertical lines 403 and 406. At 445 the client issues a DATABASE-SONGS request to obtain metadata about available songs.

[0037] A thick client may choose to issue a DATABASE-SONGS request so that it can front load network traffic. Once metadata about a song is received and stored, the client 115 does not need to request that metadata again. Playlists would only need to correctly identify a song (e.g., using the song identification number), and the client-side media management system 300 would be able to associate it with the already-received metadata.

[0038] Thin clients may choose to issue an index range, query, metadata field specifier, or skip 445 all together. A metadata field specifier would indicate that only certain metadata fields are desired. Thick clients that use 445 may also choose to use these same index range, query or metadata field specifier techniques. For example, limiting the song metadata request to only songs in a certain genre might be a technique that is used to provide the user with a different user experience.

[0039] The server 110 performs any necessary filtering operations at 448 and then issues a reply at 451. Figure 6 is a block diagram illustrating the organizational structure of the client-side media management system 300 at 453, after receiving a reply to the DATABASE-SONGS request. The song records 605 may either be a partial or complete representation of the server-side song records 215, having metadata that might, for example, include the names of songs, an identification number, a persistent identification number, the artist, the album, the size of the song, the format of the song, the required bit rate, and any other appropriate information. If the server-side management system 200 had multiple databases, a DATABASE-SONGS request (if used) would need to be issued for each database.

[0040] Figure 4D is a representational control flow diagram illustrating a technique that could be used to enumerate the playlist records 220 portion of the server-side media management system 200. Operations performed by the client 115 and the server 110 are represented by corresponding vertical lines 403 and 406.

[0041] At 454 the client issues a DATABASE-PLAYLIST request to obtain a list of available playlists. Playlists on the server 110 can either be user-identified or generated automatically by the server-side media management system 200. For example, a "base playlist" might be automatically created as a special playlist that contains all the songs in the entire song database 205 while a "John Lennon playlist" might be a user-created collection of songs by John Lennon.

[0042] After receiving the DATABASE -PLAYLIST request and performing any necessary filtering operations, the server 110 issues a reply at 457. The reply includes a list of all playlists in the music database 205 and information about those playlists. The information about the playlists might, for example, include identification numbers and/or persistent identification numbers for the playlists, and any other information (e.g., whether the songs in the playlist are in order or can be shuffled) that may have been provided. Multiple DATABASE-PLAYLIST requests may be required to populate multiple databases.

[0043] Figure 4E is a representational control flow diagram illustrating a technique that could be used to populate the playlist records 520 portion of the client-side media management system 300. Operations performed by the client 115 and the server 110 are represented by corresponding vertical lines 403 and 406. Once a playlist is identified at 460, the client 115 sends a PLAYLIST-SONGS request for that playlist at 463. Depending upon whether operations 445 through 451 were already performed, the PLAYLIST-SONGS request could additionally request that metadata accompanying each song also be delivered in order to populate the song records 605. Although thick clients that do not have a mechanism for informing the server 110 of already-received song records 605 would run the risk of receiving duplicate song records 605, thin clients that did not retain song records 605 might benefit from requesting song metadata along with each playlist.

[0044] After receiving the PLAYLIST-SONGS request and performing any necessary filtering operations, the server 110 issues a reply at 466 containing the requested information. Figure 7 is a block diagram illustrating the organizational structure of the client-side media management system 300 after receiving a reply to the PLAYLIST-SONGS request at 469. Each time the PLAYLIST-SONGS transaction between the client 115 and the server 110 is completed, another playlist record 705 is populated. The playlist record 705 may be a complete or partial representation of the corresponding server-side playlist record 220. Multiple PLAYLIST-SONGS requests may be required to populate multiple playlists.

[0045] Figure 4F is a representational control flow diagram illustrating a technique that could be used to retrieve a song from the song database 205 once a song is selected. Operations performed by the client 115 and the server 110 are represented by corresponding vertical lines 403 and 406.

[0046] At 472 the client 115 issues a SONG-DATA request to retrieve the song data from the server 110. If a single song is stored in multiple formats on the server 100, the SONG-DATA request could include a format identifier. The format identifier could specify that the song is requested in, for example, MPEG 3 encoded data ("mp3"), MPEG 4 advanced audio coding ("m4a") audio interchange file format ("aiff"), or Windows sound file ("wav"). At 475 the server 110 delivers the audio file to the client 115. In one embodiment, the server 110 will stream the song by appending song data to an http header, therefore making the client 115 responsible for parsing the data as is appropriate for playing the song at the client 115.

[0047] The preceding description assumes no mechanism is used to update the client 115 if the data on the server-side media management system 200 is modified during a session. For example, a client-side representation of a playlist 705 may not accurately represent the latest version of the corresponding server-side playlist 220.

[0048] Figure 8 is a representational control flow diagram illustrating one technique that could be used to ensure the client 115 and the server 110 are synchronized. Operations performed by the client 115 and the server 110 are represented by corresponding vertical lines 805 and 810.

[0049] At 815 the client 115 issues an UPDATE request to retrieve the media data from the server 110. The UPDATE request can be a flag that informs the server 110 that the client 115 wants to be informed when data on the server 110 (e.g., music database 205) changes. In one embodiment, the flag is a revision indicator, such as a revision number or a time-stamp.

[0050] At 820 the server 110 processes the UPDATE request. In one embodiment, the server 110 will not respond to the UPDATE request until the server's music database has changed relative to the client's representation for that music database. If a revision number was used at 815, the server 110 compares the revision number provided by the client 115 with the current revision number to determine if a change was made. The revision number of the server 110 represents the version of the music database 205 at the server 110. Any subsequent changes to records 215 or 220 may cause the server 110 to increment its revision number by one. Depending upon the requirements of the system, a group of changes to records 215 or 220 may be batched so that the revision number only increments once. Batching may be performed by standard techniques, including by operation (e.g., adding files as a group to the server), by time (e.g., waiting a certain period of time after the last change has been made to ensure no other changes will be made) or by number of operations (e.g., changing the revision number after a certain number of changes).

[0051] Once the server 110 determines that the revision number provided by the client no longer equals the current revision number, a reply is issued at 825. In one embodiment, the reply will contain the server's current revision number. Although the server 110 could then the continue to monitor for changes in the server's current revision number, the client 115 might re-issue a new UPDATE request with the server's current revision number, essentially repeating operation 815 with the updated revision number. Some systems might have the client 115 always start with a client-generated revision number of "1" and the server 110 always start with a revision number "2" in order to force the server 110 to issue an UPDATE reply the first time it receives an UPDATE request. Such an approach can provide the client 115 with an impetus for populating its local representation of the music database 205 with a SERVER-DATABASE request (see figure 4B at 433). Additionally, an UPDATE reply might also be used to notify a client 115 that the server is about to terminate connection (possibly due to timeout or server shutdown), perhaps by issuing a current revision number of 0.

[0052] In addition to the UPDATE request, the client 115 could also include a revision number field in any of the requests 433, 445, 454, and 463. The server 110 would then, on a request-by-request basis, check the revision number provided with the requests with the revision number for that particular request. If the revision numbers did not match, the server 110 would issue an UPDATE reply, specifying the current revision number and perhaps the corresponding reply. Otherwise, the server 110 would comply with the request as previously described.

[0053] In one embodiment, the database requests 433, 445, 454, and 463 would additionally support incremental updating in order to reduce network traffic (and to improve user experience through greater responsiveness). Incremental updates would allow the client to request only the changes from a historical revision number to a current revision number. If, for example, the client 115 had populated its playlists records 705 with information from revision "5," and then the client is notified by the server 110 that the latest revision is "8," the client can issue a new PLAYLIST-

SONGS request 463, requesting only the information that has changed from revision "5" to revision "8." As long as the server 110 maintains or has access to a historical record of modifications from each revision number, it would be able to comply with the incremental request.

[0054] However, an optimization might allow for the server 110 to determine whether complying with the incremental request would actually be more efficient than re-sending the entire reply. In certain circumstances (e.g., when more than half of the songs in the playlist were deleted), it would use less network resources to reply with a full PLAYLIST-SONGS reply 466 instead of an incremental reply. However, when the PLAYLIST-SONGS reply 466 provides a full reply, the reply would advantageously include an indication that the data does not represent an incremental update, so the client 115 will be able to properly handle the information.

[0055] Generally, the techniques of the present invention may be implemented on software and/or hardware. For example, they can be implemented in an operating system, in a separate user process, in a library package bound into network applications, or on a specially constructed machine. In a specific embodiment of this invention, the technique of the present invention is implemented in software such as an operating system and/or in an application program running on the operating system.

[0056] A software or software/hardware hybrid implementation of the techniques of this invention may be implemented on a general-purpose programmable machine selectively activated or reconfigured by a computer program stored in memory. In an alternative embodiment, the techniques of this invention may be implemented on a general-purpose network host machine such as a personal computer, workstation or server. Further, the invention may be at least partially implemented on a general-purpose computing device.

[0057] Referring now to FIG. 9, a computing device 900 suitable for implementing the techniques of the present invention includes a master central processing unit (CPU) 905, interfaces 910, memory 915 and a bus 920. When acting under the control of appropriate software or firmware, the CPU 905 may be responsible for implementing specific functions associated with the functions of a desired computing device. The CPU 905 preferably accomplishes all these functions under the control of software including an operating system (e.g., Mac OS X), and any appropriate applications software (e.g., iTunes).

[0058] CPU 905 may include one or more processors such as those from the Motorola family of microprocessors or the MIPS family of microprocessors. In an alternative embodiment, the processor is specially designed hardware for controlling the operations of computing device 900.

[0059] The interfaces 910 are typically provided as interface cards. Generally, they control the sending and receiving of data packets over the network and sometimes support other peripherals used with the computing device 900. Among the interfaces that may be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, and the like. In addition, various very high-speed interfaces may be provided such as fast Ethernet interfaces, Gigabit Ethernet interfaces, ATM interfaces, HSSI interfaces, POS interfaces, FDDI interfaces, ASI interfaces, DHEI interfaces and the like. Generally, these interfaces may include ports appropriate for communication with the appropriate media. In some cases, they may also include an independent processor and, in some instances, volatile RAM.

[0060] Regardless of computing device's configuration, it may employ one or more memories or memory modules (such as, for example, the memory 915) configured to store data, program instructions and/or other information relating to the functionality of the techniques described herein. The program instructions may control the operation of an operating system and/or one or more applications, for example.

[0061] Because such information and program instructions may be employed to implement the systems/methods described herein, the present invention relates to machine readable media that include program instructions, state information, etc. for performing various operations described herein. Examples of machine-readable media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media such as floptical disks; and hardware devices that are specially configured to store program instructions, such as read-only memory devices (ROM) and random access memory (RAM). The invention may also be embodied in a carrier wave traveling over an appropriate medium such as airwaves, optical lines, electric lines, etc. Examples of program instructions include both machine code, such as produced by a compiler, and higher level code that may be executed by the computer (e.g., using an interpreter).

[0062] Although illustrative embodiments and applications of this invention are shown and described herein, many variations and modifications are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those of ordinary skill in the art after perusal of this application. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.